RELATÓRIO TÉCNICO
RT – ES XXX / 2007

Revisão *quasi*-Sistemática da Literatura:

Caracterização de Métodos Ágeis de Desenvolvimento de Software

José Fortuna Abrantes

Guilherme Horta Travassos

Universidade Federal do Rio de Janeiro – COPPE/Sistemas
Caixa Postal 68.511 – CEP 21.941-972 – Rio de Janeiro – RJ – Brasil

{jfa, ght}@cos.ufrj.br

PESC
Programa de Engenharia de Sistemas e Computação
COPPE / UFRJ

Rio de Janeiro, dezembro de 2007

Revisão *quasi*-Sistemática da Literatura:

Caracterização de Métodos Ágeis de Desenvolvimento de Software

José Fortuna Abrantes

Guilherme Horta Travassos

Universidade Federal do Rio de Janeiro – COPPE/Sistemas
Caixa Postal 68.511 – CEP 21.941-972 – Rio de Janeiro – RJ – Brasil

{jfa, ght}@cos.ufrj.br

ABSTRACT

This technical report aims to describe an investigation concerned with the characteristics that could be used to characterize agile development scenarios. A research protocol was formalized and executed in order to conduct a *quasi*-systematic review, which means that no comparison could be possible since it represents a characterization review. The searches were conducted in the period of November-December/2006. The data obtained from the technical literature were analyzed and have allowed observing the characteristics that could be used to characterize a software development method as agile.

RESUMO

Este relatório técnico tem por objetivo descrever uma pesquisa relacionada a identificação de características de agilidade no contexto ddo desenvolvimento de software. Um protocolo de pesquisa foi formalizado e executado para conduzir uma *quasi*-revisão sistemática da literatura, o que significa que não pode ser possível executar qualquer comparação desde que esta representa uma revisão de caracterização. As buscas foram realizadas no período de novembro-dezembro de 2006. Os dados obtidos da literatura técnica foram analisados e permitiram observar as características que poderiam ser utilizadas para caracterizar um método de desenvolvimento de software como sendo ágil.

1- Introdução

A redução do ciclo de desenvolvimento foi considerada uma das principais metas de desenvolvimento de software, a partir da década de 1990. Neste cenário, Aoyama (1998) definiu agilidade em processos de software como sendo a capacidade de adaptações rápidas a mudanças, nos requisitos e no ambiente que envolve o software. Aoyama propôs um processo ágil a partir de experiências com desenvolvimento concorrente e distribuído, de lições aprendidas em fábricas de software japonesas e de conceitos em processos de produção de hardware. Foi então "cunhada" a idéia de processo ágil de software, que não significa simplesmente desenvolvimento rápido de aplicações, mas principalmente capacidade de adaptação com rapidez e flexibilidade a mudanças nos processos, nos produtos e no ambiente.

Nos últimos anos, os métodos ágeis têm se apresentado como uma alternativa para o desenvolvimento de software. Também pode ser observado o crescimento do interesse das organizações desenvolvedoras de software por tais métodos. Para Boehm e Turner (2003, 2004) as metodologias ágeis de desenvolvimento prometem alto grau de satisfação do cliente, menores taxas de defeitos, tempos de desenvolvimento mais rápidos e uma solução para mudanças rápidas de requisitos. Os métodos ágeis de desenvolvimento de software têm chamado a atenção de engenheiros de software e de pesquisadores em todo o mundo [Abrahamsson, et al. 2003]. Segundo Hazzan e Dubinsky (2006) desenvolvimento ágil de software tem a ver com mudanças culturais nos ambientes de desenvolvimento de software.

Para Abrahamsson, Salo, Ronkainen e Warsta (2002) o que os métodos ágeis buscam não é como conter as mudanças mais cedo em um projeto de software, mas a melhor maneira de tratar mudanças inevitáveis ao longo de seu ciclo de vida. Para alcançar seu objetivo, os métodos ágeis são projetados para (1) produzir a primeira entrega em semanas e alcançar realimentação (*feedback)* rápida e mais cedo; (2) criar soluções mais simples de modo que se houver mudanças que haja mais facilidade e menor volume de alterações a serem feitas; (3) melhorar continuamente a qualidade do projeto, fazendo com que a iteração seguinte tenha menor custo de implementação; (4) testar constantemente, para detectar defeitos mais cedo e removê-los com menor custo.

O entendimento do significado de agilidade no contexto de métodos ágeis é importante para guiar a evolução de idéias e pesquisas relacionadas com práticas de teste de software compatíveis com os métodos ágeis. Tal entendimento poderá apoiar também, trabalhos relacionados com a verificação do alinhamento de características de projetos de software e alternativas de métodos ágeis para desenvolvê-los.

A crescente imprevisibilidade do futuro é um dos aspectos mais desafiadores da nova economia. Turbulências tanto nos negócios quanto na tecnologia causam mudanças, que podem ser vistas como ameaças das quais se defender, ou como oportunidades a serem abraçadas. A abordagem ágil, ao invés de resistir à mudança, busca acomodá-la o mais fácil e eficientemente possível, mantendo consciência de suas possíveis consequências. A maioria das pessoas concordam que realimentação *(feedback)* é importante, mas esquecem que o resultado de sua aceitação é a mudança [Fowler e Highsmith, 2001].

Segundo Sugden e Strens (1996), para minimizar o impacto negativo da mudança nos objetivos do projeto e maximizar seus benefícios, três principais

estratégias são reconhecidas: identificar a mudança mais cedo no ciclo de vida; facilitar a incorporação da mudança quando inevitável e se possível reduzir a incidência de necessidades de mudança.

Para Rajlich (2006) a tentativa de congelar os requisitos por toda a duração do projeto não funciona, porque a acumulação de mudanças, tanto internas quanto externas é muito grande e resultaria em falhas no projeto. O extremo oposto (afogar os programadores com um fluxo constante de mudanças) também não funciona porque criaria o caos no projeto. Um compromisso razoável seria congelar o ambiente por um tempo limitado chamado iteração. No final de cada iteração os *stakeholders* avaliam o progresso, consideram as mudanças no ambiente e decidem os caminhos da próxima iteração. As iterações fornecem realimentação (*feedback)* e reduzem os riscos de surpresas e falhas. Iterações rápidas são recomendadas por processos ágeis.

Há exemplos no mundo real, de argumentações a favor e contra os métodos ágeis [Boehm, 2002]. De todo modo, torna-se relevante conhecer mais sobre atividades de processos de desenvolvimento de software dentro do contexto de métodos ágeis de desenvolvimento. Do mesmo modo, identificar características de projetos de software que possam credenciá-los como candidatos a aplicação de métodos ágeis, bem como identificar características de métodos ágeis que os credencie para aplicação em determinados projetos de software, é uma tarefa importante.

Abrahamsson et al. (2003) considera relevante entender as propriedades de cada método ágil em particular, para que se possa fazer um julgamento adequado quando for o caso. Para analisar cada método ágil podem-se adotar perspectivas. E para caracterizar desenvolvimento ágil de software em geral, é necessário identificar atributos.

Boehm e Turner (2004) entendem que a agilidade responde pela liberdade e criatividade em um projeto de software, aplicando memória e história para se ajustar a novos ambientes, para reagir e se adaptar, tirando vantagens de oportunidades inesperadas e para atualizar uma base de experiências para o futuro.

O Longman Dictionary of Contemporary English (1992) define “ágil” como sendo a qualidade de ser capaz de se mover rápida e facilmente. O Dicionário Lello Popular (1958) define ágil como a qualidade do que está vivo e se move com facilidade e presteza. O Dicionário Aurélio, Holanda (1977), define ágil como hábil, o que se move com destreza, que tem presteza de movimentos, ligeiro.

Não há definição universalmente aceita para um método ágil no campo de desenvolvimento de sistemas de informação [Conboy e Fitzgerald, 2004].

Nesse contexto, é interessante identificar e observar quais propriedades ou características devem estar associadas a um método de desenvolvimento de software, para que ele possa ser considerado ágil. Métodos ágeis não são *ad-hoc* e requerem disciplina por parte das equipes que os utilizam [Taylor et al, 2006].

A questão básica de pesquisa é determinar o que caracteriza um método de desenvolvimento de software como sendo um método ágil. Pretende-se chegar a um conjunto básico de características que são necessárias para que um método possa ser classificado como método ágil, investigando através de revisão sistemática de literatura (estudo secundário) [Biolchini et al., 2005], quais são as características de agilidade no contexto de métodos ágeis. O que significa “ser ágil” para um método de desenvolvimento de software?

Será adotada uma abordagem que estrutura a questão de pesquisa em 4 elementos básicos: população, intervenção, comparação e resultado [Pai et al, 2004]. Tendo em vista ser o objetivo deste estudo realizar uma caracterização da área, não haverá comparação e nem será possível aplicação de meta-análise. Desta forma, podemos definir este tipo de estudo secundário, apesar de sistemático, como uma *quasi*-revisão sistemática.

A motivação desta pesquisa é servir de apoio à busca de entendimento e solução a outras questões envolvendo atividades de processo com métodos ágeis.

Este trabalho se estrutura da seguinte forma: na seção 2 é explicitado o objetivo do trabalho. Na seção 3 é apresentado um protocolo elaborado especificamente para esta revisão sistemática. Na seção 4 descreve-se a execução do protocolo apresentado na seção 3. Na seção 5 são apresentados os resultados obtidos com a revisão sistemática. Na seção 6 apresenta-se uma proposta de caracterização básica de métodos ágeis de desenvolvimento de software. Na seção 7 são apresentadas as conclusões deste trabalho e sugestões para seus possíveis desdobramentos. Na seção 8 estão registradas as referências bibliográficas.

2- Objetivo do Trabalho

Pretende-se investigar, através de revisão sistemática de literatura, quais são as propriedades ou características de agilidade no contexto de métodos ágeis para desenvolvimento de software. O que significa "ser ágil" para um método de desenvolvimento de software?

Não se pretende investigar características de métodos ágeis específicos, mas de uma maneira geral, identificar quais são as propriedades ou características desse grupo de métodos para desenvolver software e assim obter um conjunto de características desejáveis para um método ser considerado ágil.

3- Protocolo de Revisão Sistemática de Literatura

Será adotada uma abordagem que estrutura a questão de pesquisa em quatro elementos básicos: população, intervenção, comparação e resultado [Pai et al, 2004].

3.1. Objetivo

Identificar as propriedades ou características dos métodos ágeis de desenvolvimento de software, de uma maneira geral.

3.2. Formulação da Pergunta

Quais são as propriedades ou características dos métodos ágeis de desenvolvimento de software?

Problema:
Encontrar propriedades ou características de métodos ágeis de desenvolvimento de software.

Aplicação:
Servir de base ou apoiar pesquisas envolvendo (1) atividades de processo em métodos ágeis de desenvolvimento de software e envolvendo (2) critérios para seleção de métodos ágeis a serem aplicados em projetos de desenvolvimento de software.

População:
Projetos de desenvolvimento de software.

Intervenção:
Métodos ágeis de desenvolvimento de software.

Comparação:
Não há.

Resultado:
Lista de propriedades ou características de agilidade de métodos de desenvolvimento de software.

Controle:
Miller, G. G. The characteristics of agile software processes, The 39th International Conference of Object-Oriented Languages and Systems (TOOLS 39), Santa Barbara, CA, 2001.

Abrahamsson, P. Salo, O. Ronkainen, J. Warsta, J. Agile Software Development Methods. Review and Analysis. Espoo. VTT Publications 478, 2002.

Lindvall, M. Basili, V. et al. Empirical Findings in Agile Methods, In: Proceedings of Extreme Programming and Agile Methods – SP/Agile Universe, pp. 197-207, 2002.

3.3. Seleção de Fontes

As fontes serão bases de dados eletrônicas, disponíveis no portal CAPES, incluindo conferências, *journals* e relatórios técnicos indexados por:

Compendex EI
IeeeXplore
Inspec
Web of Science
ACM digital library

Estas fontes foram escolhidas porque são as que se tem acesso para recuperação de referências, bem como maior facilidade para recuperação do texto completo do artigo quando fosse o caso. Além disso, estas fontes foram consideradas significativas no sentido de oferecerem publicações pertinentes e que podem contribuir significativamente para o resultado da pesquisa.

Idioma:
Inglês, por ser maioria nas bases de dados pesquisadas. Em uma busca preliminar, não foram encontrados artigos escritos em português que trouxessem contribuição para o tema da pesquisa. Além disso, textos em português, embora se reconheça a sua importância, muitas vezes não se encontram indexados, o que aumenta o esforço ou impede sua busca.

Tipos de documentos:
Qualquer tipo de trabalho ou artigo que faça abordagem sobre propriedades ou características de agilidade em métodos de desenvolvimento de software.

3.4. Palavras-chave

População:
software, development, project, system, application, engineering, building, implementation

Intervenção:
agile, method, adaptive, rapid, approach, technique, environment, process, practice, methodology

Resultado:
characteristic, attribute, property, feature, characterization, aspect, idea, factor, dimension, driver, perspective, requirement

3.5. Critérios de Inclusão e Exclusão

Os documentos devem estar disponíveis na web;

Os documentos devem contemplar propriedades ou características de agilidade em métodos ágeis de desenvolvimento de software;

Os documentos devem relatar algum exemplo ou caso de utilização de algum método ágil de desenvolvimento de software.

3.6. Processo de Seleção dos Estudos

O pesquisador aplicará a estratégia de busca para a identificação de potenciais documentos. Os documentos identificados serão selecionados pelo mesmo pesquisador através da leitura e verificação dos critérios de inclusão e exclusão estabelecidos.

Posteriormente a lista de documentos excluídos será avaliada por um segundo pesquisador. Em caso de conflito o documento será incluído.

Ao final, os documentos serão lidos pelos pesquisadores para extração de informações sobre propriedades ou características dos métodos ágeis de desenvolvimento de software.

3.7. Avaliação da Qualidade dos Estudos

Não será feita por tratar-se de uma pesquisa para fins de caracterização de objeto de estudo (os métodos ágeis). Será considerado que as fontes dos documentos são confiáveis, e que os textos tenham passado por revisões externas que serviram de filtragem para que tenham qualidade suficiente para contribuir com a revisão sistemática.

3.8. Estratégia de Extração de Informações

Para cada estudo selecionado após a execução do processo de seleção, serão extraidas as seguintes informações:

- Título do documento
- Autor(es)
- Fonte
- Ano de publicação
- Propriedades ou características de agilidade

3.9. Sumarização de Resultados

Os resultados serão tabulados. Serão realizadas análises para identificar similaridades e as propriedades ou características mais relevantes para os métodos ágeis de desenvolvimento de software. Será considerada a freqüência com que uma propriedade ou característica tenha sido apontada por autores diferentes.

3.10. *String* de Busca

Na medida do possível, a *string* de busca será a mesma para todas as máquinas de busca. Contudo, poderá haver adaptações para se adequar a restrições de máquinas de busca específicas, observando-se as seguintes diretrizes:

1. a *string* derivada deverá ser logicamente equivalente à *string* original, ou

2. na impossibilidade de se manter equivalência exata, deverá a *string* derivada ser mais abrangente para evitar perda de documentos potencialmente relevantes.

De acordo com Pai et al (2004) os 4 elementos básicos que estruturam a questão de pesquisa podem ser relacionados com o operador lógico AND. Isto foi feito para cada conjunto de palavras-chave escolhidas para representar cada um dos elementos "população", "intervenção" e "resultado". Para cada um destes três elementos da estrutura, as respectivas palavras-chave foram combinadas com o operador lógico OR. Foi também utilizado o operador binário NEAR, que é um recurso oferecido pelas máquinas de busca para estabelecer que uma palavra-chave deva ocorrer próxima à outra palavra-chave. Este operador pode ser parametrizado para estabelecer a proximidade ou distância em quantidade de palavras, entre as duas palavras-chave, em cada ocorrência. Se utilizado o parâmetro zero, as duas palavras-chave devem ocorrer juntas para satisfazer a condição.

Segue-se a *string* de base:

(software NEAR0 development OR software NEAR1 engineering OR software NEAR0 building OR software NEAR0 implementation OR software NEAR0 projects OR software NEAR0 systems OR software NEAR0 application OR system NEAR0 development OR system NEAR0 engineering OR system NEAR0 building OR system NEAR0 implementation OR system NEAR0 project OR application NEAR0 development OR application NEAR0 engineering OR application NEAR0 building OR application NEAR0 implementation OR application NEAR0 project) **AND** ((agile OR adaptive OR rapid) AND (method OR process OR practice OR methodology OR approach OR technique OR environment)) **AND** ((agile) AND (characteristic OR attribute OR property OR feature OR characterization OR aspect OR idea OR factor OR dimension OR driver OR perspective OR requirement))

4- Execução de Buscas

As buscas foram realizadas utilizando máquinas de busca de editoras ou bibliotecas digitais disponíveis no portal CAPES.

4.1. Avaliação das *Strings* de Busca

A busca, em todas as máquinas, com exceção da ACM digital library, foram efetuadas com a opção *autostemming* ligada e com utilização do operador NEAR parametrizado.

Os resultados obtidos com a *string* básica foram comparados com os resultados obtidos em uma busca feita anteriormente, utilizando uma *string* menos abrangente.

Observou-se que algumas máquinas de busca (Inspec, Web of Science), apesar de terem submetidas a si, uma *string* mais abrangente, deixaram de recuperar documentos relevantes, recuperados anteriormente. Por este motivo, foram feitos ajustes nas *strings*, até produzir resultado satisfatório, recuperando quantidade de documentos pelo menos igual à busca menos abrangente feita antes e não perdendo os documentos relevantes também recuperados anteriormente. As alterações realizadas para cada máquina de busca serão apresentadas a seguir:

Compendex EI:

a busca não apresentou problemas, a *string* foi processada integralmente e, pelos resultados obtidos, não precisou ser ajustada.

(software NEAR/0 development OR software NEAR/1 engineering OR software NEAR/0 building OR software NEAR/0 implementation OR software NEAR/0 projects OR software NEAR/0 systems OR software NEAR/0 application OR system NEAR/0 development OR system NEAR/0 engineering OR system NEAR/0 building OR system NEAR/0 implementation OR system NEAR/0 project OR application NEAR/0 development OR application NEAR/0 engineering OR application NEAR/0 building OR application NEAR/0 implementation OR application NEAR/0 project) AND ((agile OR adaptive OR rapid) **AND** (method OR process OR practice OR methodology OR approach OR technique OR environment)) **AND** ((agile) AND (characteristic OR attribute OR property OR feature OR characterization OR aspect OR idea OR factor OR dimension OR driver OR perspective OR requirement))

IeeeXplore:

a busca apresentou problemas. A *string* não foi processada integralmente e teve que ser feita por partes, sucessivamente em cima dos resultados parciais.

Observou-se que na última etapa a máquina não conseguiu processar a consulta, mesmo sendo sobre os resultados parciais alcançados.

Seguindo as diretrizes estabelecidas no protocolo, a *string* foi reformulada e teve que ficar mais abrangente. Foram então consideradas para busca apenas a intervenção e o resultado, desconsiderando-se a população e absorvendo-se o ônus de selecionar documentos manualmente.

A string original (não processada pela máquina) é a que se segue:

(software NEAR/0 development OR software NEAR/1 engineering OR software NEAR/0 building OR software NEAR/0 implementation OR software NEAR/0 projects OR software NEAR/0 systems OR software NEAR/0 application OR system NEAR/0 development OR system NEAR/0 engineering OR system NEAR/0 building OR system NEAR/0 implementation OR system NEAR/0 project OR application NEAR/0 development OR application NEAR/0 engineering OR application NEAR/0 building

OR application NEAR/0 implementation OR application NEAR/0 project) **AND** ((agile OR adaptive OR rapid) AND (method OR process OR practice OR methodology OR approach OR technique OR environment)) **AND** ((agile) AND (characteristic OR attribute OR property OR feature OR characterization OR aspect OR idea OR factor OR dimension OR driver OR perspective OR requirement))

Tendo em vista a impossibilidade de se processar a *string* completa, foi feito um desmembramento da mesma, tomando-se o cuidado de tentar um esquema que minimizasse a quantidade de consultas intermediárias. Para isso buscou-se determinar a extensão máxima da string que poderia ser processada pela máquina de busca da IeeeXplore, e a partir daí foram montadas as *strings* complementares. A partir de uma *string* base foram montadas 11 *strings* diferentes cujos resultados foram agrupados. As repetições foram determinadas e eliminadas. O processamento do resultado seguiu o mesmo padrão adotado para todas as máquinas das demais editoras ou bibliotecas digitais.

A *string* de base utilizada foi a que se segue:

(software <NEAR/0> develop OR software <NEAR/1> engineering OR software <NEAR/0> building OR software <NEAR/0> implement OR software <NEAR/0> projects OR software <NEAR/0> systems OR software <NEAR/0> application OR system <NEAR/0> develop OR system <NEAR/0> engineering OR system <NEAR/0> building OR system <NEAR/0> implement OR system <NEAR/0> project OR application <NEAR/0> develop OR application <NEAR/0> engineering OR application <NEAR/0> building OR application <NEAR/0> implement OR application <NEAR/0> project) AND (agile OR adaptive OR rapid) AND (method OR process OR practice OR methodology OR approach OR technique OR environment) AND agile

Em seguida, foram formadas 11 strings, acrescentando-se à *string* base o operador AND e cada um do seguintes termos: charact*, attribute, propert*, feature, aspect, idea, factor, dimension, driver, perspective e requirement. Foram efetuadas portanto, 11 buscas na máquina da IeeeXplore.

Inspec:

A máquina de busca conseguiu processar a *string* completa. Contudo, recuperou menor quantidade de documentos do que a busca com a *string* menos abrangente efetuada anteriormente, e, além disso, deixou de recuperar um controle anteriormente recuperado. Por este motivo a *string* teve que ser reformulada.

Observou-se através de testes que o problema não estava no operador NEAR, mas sim no termo “characteristic” que teve que ser passado para o plural, fazendo com que o controle fosse recuperado.

Tendo em vista este resultado, todos os nomes na *string* base foram passados para o plural e a máquina de busca passou a recuperar o controle, os documentos relevantes da busca com a string menos abrangente feita anteriormente e passou a recuperar 250 documentos ao invés de apenas 77.

A *string* final para a máquina da Inspec ficou assim:

(software NEAR0 development OR software NEAR1 engineering OR software NEAR0 building OR software NEAR0 implementation OR software NEAR0 projects OR software NEAR0 systems OR software NEAR0 applications OR systems NEAR0 development OR systems NEAR0 engineering OR systems NEAR0 building OR systems NEAR0 implementation OR systems NEAR0 projects OR applications NEAR0 development OR applications NEAR0 engineering OR applications NEAR0 building OR applications NEAR0 implementation OR applications NEAR0 projects) AND ((agile OR adaptive OR rapid) AND (methods OR processes OR practices OR methodologies OR approaches OR techniques OR environments)) AND ((agile) AND (characteristics OR attributes OR properties OR features OR characterization OR aspects OR ideas OR factors OR dimensions OR drivers OR perspectives OR requirements))

Na máquina da Inspec, a sintaxe do operador NEAR é diferente da sintaxe do mesmo operador nas máquinas Compendex EI e IeeeXplore.

Além disso, na máquina da Inspec, a *string* completa só pode ser processada corretamente na opção "*Normal Search*".

Web of Science:

Na máquina da Web of Science o operador equivalente ao NEAR das máquinas anteriores é o SAME que não pode ser parametrizado.

A máquina de busca não conseguiu processar a string completa. Por limitações de implementação ficou restrita a 50 termos de busca e 49 operadores.

Sendo assim, foram suprimidos da *string* básica, 7 termos de busca, a saber: (*idea, software building, system building, application building*). Estes termos foram escolhidos porque são sinônimos menos específicos e mais abrangentes, incluídos no conjunto com o intuito de aumentar o *recall*. A máquina recuperou 23 documentos, acima dos 11 recuperados em busca anterior com *string* menos abrangente.

Contudo, continuou deixando de recuperar documentos recuperados com a *string* menos abrangente em busca anterior. Sendo assim, foi feita a passagem para o plural, de todos os nomes usados na *string*, como já havia sido feito para a Inspec. O resultado foi bom, sendo recuperados os documentos relevantes anteriores e a quantidade total de itens recuperados pela máquina passou a ser de 45 documentos.

A opção de busca utilizada na Web of Science teve que ser a "*General Search*".

A *string* final utilizada para a Web of Science ficou assim:

(software SAME development OR software SAME engineering OR software SAME implementation OR software SAME projects OR software SAME systems OR software SAME application OR systems SAME development OR systems SAME engineering OR systems SAME implementation OR systems SAME projects OR applications SAME development OR applications SAME engineering OR applications SAME implementation OR applications SAME projects) **AND** ((agile OR adaptive OR rapid) AND (methods OR processes OR practices OR methodologies OR approaches OR techniques OR environments)) **AND** ((agile) AND (characteristics OR attributes OR properties OR features OR characterization OR aspects OR factors OR dimensions OR drivers OR perspectives OR requirements))

ACM digital library:

Por ter uma máquina de busca mais restrita, que não permite o uso de operadores booleanos, a *string* básica teve que ser reformulada para se adaptar aos operadores da ACM, observando-se, contudo, as diretrizes estabelecidas no protocolo.

A opção de busca utilizada na ACM *digital library* foi a "*Advanced Search*", com a *digital library* marcada.

A *string* final utilizada para a ACM *digital library* ficou assim:

+agile +software +development +method* +process* +characteristic* +attribute* propert* feature* characterization aspect* idea* factor* dimension* driver* perspective* requirement* adaptive rapid characterization engineering building implementation project* system* application* practice* approache* technique* environment*

4.2. Execução das Buscas nas Bibliotecas Digitais

Resultados:

A quantidade de referências recuperadas foi:

Biblioteca	Quantidade Recuperada
Compendex EI	303
IeeeXplore	299
Inspec	250
Web of Science	45
ACM *digital library*	119
TOTAL	1016

4.3. Análise dos Documentos Recuperados

Todas as referências recuperadas foram importadas para o JabRef [JabRef version 2.0.1 (c) 2006] que foi o gerenciador de referências utilizado para manipular as referências recuperadas pelas máquinas de busca. A ferramenta JabRef foi muito útil para identificar repetições, categorizar referências, priorizar leituras, tabular referências, exportar listas para recuperação de documento completo, etc. Foram criados campos customizados na ferramenta JabRef, bem como a elaboração de layouts específicos de exportação, de modo a facilitar o trabalho de manipulação das referências, inclusive

para priorizar a leitura dos documentos incluídos e selecionados para a revisão sistemática.

Nem todas as editoras / bibliotecas digitais permitem as mesmas facilidades para exportação do material recuperado em formato que possa ser importado por um gerenciador de referências. A Compendex EI, a Inspec, e a Web of Science exportam de uma única vez todos os itens recuperados pelas respectivas máquinas. A Inspec e Web of Science exportaram referências com muitos campos faltantes (não preenchidos). A IeeeXplore só permite exportação por página de resultado. Quando se tentou reduzir a quantidade de exportações aumentando a quantidade de itens por página, a exportação não funcionou, só podendo ser feita para uma das quatro opções de quantidade por página de exibição oferecida pela editora, que é exatamente a mínima, com 25 itens recuperados, embora a máquina ofereça opção para exibir até 100 itens por página. A ACM *digital library* só permite exportação item a item o que dificulta bastante o tratamento dos itens recuperados.

As repetições foram eliminadas, mantendo-se o artigo remanescente contabilizado para a biblioteca digital com maior quantidade de itens recuperados. Todos os controles foram recuperados (a Compendex EI recuperou dois deles e a Inspec recuperou o outro).

A nova situação quantitativa, após eliminação de repetiçõess, ficou assim:

Biblioteca	**Quantidade Inicial**	**Artigos Repetidos**	**Artigos Mantidos**
Compendex EI	303	2	301
IeeeXplore	299	86	213
Inspec	250	126	124
Web of Science	45	28	17
ACM digital library	119	2	117
TOTAL	1016	244	772

Em seguida, em uma primeira avaliação superficial (título e *abstract*) foram excluídas as referências que nitidamente tratavam de outros assuntos não pertinentes à pesquisa. Após eliminação também de tais itens, a nova situação quantitativa ficou assim:

Biblioteca	A serem Avaliados	Excluídos	Selecionados Inicialmente
Compendex EI	301	53	248
IeeeXplore	213	12	201
Inspec	124	6	118
Web of Science	17	8	9
ACM digital library	117	102	15
TOTAL	772	181	591

Posteriormente, em uma avaliação mais apurada e detalhada, foram selecionados os documentos candidatos a fazer parte da revisão sistemática. Para aproveitar as recuperações já efetuadas, e enriquecer a revisão, foram separados documentos relevantes que tratam de agilidade de processos em outras áreas de conhecimento (manufatura/negócios e eletrônica) com o objetivo de apurar se alguma idéia foi ou pode ser aproveitada para o processo de desenvolvimento de software.

Após a última seleção os quantitativos ficaram assim:

Biblioteca	Selecionados Inicialmente	Artigos Incluidos				Artigos Excluidos			
		TOT	Software	Neg/Manuf	Eletrônica	TOT	Software	Neg/Manuf	Eletrônica
Compendex EI	248	86	70	15	1	162	133	25	4
IeeeXplore	201	113	85	23	5	88	34	32	22
Inspec	118	42	39	3		76	70	6	
Web of Science	9	4	3	1		5	3	2	
ACM digital library	15	7	7			8	8		
TOTAL	**591**	**252**	**204**	**42**	**6**	**339**	**248**	**65**	**26**

Do total de documentos recuperados, 23,9% eram repetições, em sua grande maioria causadas por superposição de recuperações efetuadas nas diferentes bibliotecas digitais. Eliminadas as repetições, 23,3% deste subtotal de documentos (181 em 772) não foram considerados relevantes. Após o exame minucioso dos documentos a partir de título e resumo, mantida a mesma base (772 documentos) 32,8% dos documentos recuperados (252 documentos) foram incluidos e selecionados para leitura. Dentre os documentos excluídos, a grande maioria (73,1%) trata de métodos ágeis, sem, contudo, abordar suas propriedades ou características.

Os 204 documentos incluidos e pertinentes à área de software foram classificados em uma escala de prioridades, p0, p1 e p2, com base na leitura minuciosa de título e resumo. Isto foi feito para permitir um critério no seqüenciamento das leituras. Em p0 foram classificados os documentos identificados como tendo alta probabilidade de relevância para a pesquisa. Em p2 foram classificados os documentos com pouca probabilidade de serem relevantes para a pesquisa. Em p1 foram classificados os documentos que geravam dúvidas quanto à sua melhor classificação considerando-se apenas p0 e p2. Essa categorização resultou na seguinte tabela quantitativa:

p0	58 documentos
p1	72 documentos
p2	74 documentos

Dos 252 documentos a serem lidos, poderiam ser identificados como prioritários, 106 documentos, com a seguinte distribuição:

Eletrônica	6
Negócios/Manufatura	42
Métodos Ágeis de Desenvolvimento de Software	58
TOTAL	106

4.4. Extração de Informações

Das 58 referências da área de software, 39 tiveram o texto completo recuperado. Estes documentos foram lidos e analisados para a revisão sistemática. Não foi possível Ter acesso ao texto completo de 19 destas referências.

Das 42 referências das áreas de negócios/manufatura, 29 tiveram o texto completo recuperado. Estes textos foram lidos e analisados para verificar se havia alguma contribuição que pudesse ser aproveitada na revisão sistemática. Não foi possível ter acesso ao texto completo de 13 destas referências.

A grande maioria dos documentos lidos não trazia uma contribuição direta para caracterizar métodos ágeis de desenvolvimento de software. Muitos deles mencionavam alguma característica, mas não traziam a semântica ou pelo menos a idéia principal para as características citadas nos textos.

Contribuíram para a revisão sistemática e tiveram as caracterizações aproveitadas, os textos de 11 documentos apenas. Dentre eles, considerou-se 2 *clusters* de documentos, sendo cada *cluster* computado uma única vez na incidência das características nos artigos, para evitar a repetição de contagem de características similares.

Na formação dos *clusters* para fins de contagem, foram utilizados os seguintes critérios: não contar repetições em artigos diferentes do mesmo autor. Também não contar repetições, quando o artigo foi escrito por mais de um autor, e as mesmas características foram abordadas por algum dos co-autores em outro artigo. Do mesmo modo, não contar repetições, quando características foram abordadas em um artigo, referenciando outro artigo no qual elas também já foram consideradas.

As propriedades ou características de métodos ágeis de desenvolvimento de software capturadas nos documentos incluídos foram tabeladas conforme se segue:

Tabela 1 – Documentos descrevendo características de métodos ágeis

Ord		
1	**Autor, Ano de publicação**	Abrahamsson, P.; Warsta, J.; Siponen, M.T. & Ronkainen, J. 2003
	Titulo	New directions on agile methods: a comparative analysis.
	Fonte	IEEE Computer Society
	Propriedades ou	**"Incrementalidade"**: pequenas *releases* de software, com ciclos rápidos de desenvolvimento;

	Características	**Transparência ou Clareza**: o método é fácil de aprender e modificar e é suficientemente documentado. **Adaptabilidade**: habilidade de atender e reagir a mudanças de última hora. **"Cooperatividade"**: interação de perto entre desenvolvedor e cliente.
2	**Autor, Ano de publicação**	Boehm, B. & Turner, R. 2004a
	Titulo	New directions on agile methods: a comparative analysis.
	Fonte	IEEE Computer Society
	Propriedades ou Características	**"Incrementalidade"**: pequenas *releases* de software, com ciclos rápidos de desenvolvimento; **Transparência ou Clareza**: o método é fácil de aprender e modificar e é suficientemente documentado. **Adaptabilidade**: habilidade de atender e reagir a mudanças de última hora. **"Cooperatividade"**: interação de perto entre desenvolvedor e cliente.
3	**Autor, Ano de publicação**	Meso, Peter. Jain, Radhika. summer 2006.
	Titulo	Contemporary practices in systems development. Agile Software Development: adaptive systems principles and best practices.
	Fonte	Information Systems Management
	Propriedades ou Características	**Princípio dos sistemas abertos**: interação aberta entre os vários *stakeholders* (gerentes, desenvolvedores e clientes); **Princípio de estrutura "enxuta" ou leve (*lean*)**: equipes capazes de procurar e receber continuamente, realimentação (*feedback)* de modo mais freqüente e com mais rapidez. (teoria dos sistemas adaptativos complexos)
4	**Autor, Ano de publicação**	Holmstrom, Helena. Fitzgerald, Brian. et al. summer 2006.
	Titulo	Contemporary practices in systems development. Agile Practices Reduce Distance in Global Software Development.
	Fonte	Information Systems Management
	Propriedades ou Características	**Ciclos de desenvolvimento iterativos**, curtos, dirigidos por características do produto; **Períodos de reflexão e introspecção**; **Tomada de decisão colaborativa**; **Incorporação de** realimentação (*feedback)* **rápida**; **Integração contínua de mudanças de código ao sistema em desenvolvimento**.
5	**Autor, Ano de publicação**	Miller, Granville G. 2001.
	Titulo	The Characteristics of Agile Software Processes.
	Fonte	Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01)
	Propriedades ou Características	**Modularidade**: característica que permite que um processo seja quebrado em componentes chamados de atividades; **Iteratividade**: foco em ciclos curtos, nos quais certo conjunto de atividades é completado em poucas semanas; estes ciclos não serão

		suficientes para obter os elementos 100% prontos, por isso são repetidos muitas vezes para refinar as entregas. **Limitação de tempo**: pode-se estabelecer limite de tempo para cada iteração (é a unidade para planejamento do projeto de desenvolvimento de software) programando-as; provavelmente não será possível programar todas as atividades do processo numa iteração, mas apenas aquelas para alcançar os objetivos estabelecidos no início da iteração; funcionalidades podem ser reduzidas e atividades reprogramadas se não puderem ser completadas no espaço de tempo alocado para aquela iteração. **Parcimônia**: ("enxutez" ou leveza - *leanness*) característica que o processo ágil tem de requerer o mínimo necessário de atividades para mitigar riscos e alcançar metas; a redução de atividades permitiria aos desenvolvedores entregar sistemas com cronogramas agressivos. **Adaptabilidade**: adaptação do processo para atender situações ou riscos não previstos inicialmente; se não puderem ser atendidas com as atividades planejadas para a iteração, novas atividades são adicionadas à iteração para alcançar as metas, ou atividades são descartadas da iteração se riscos inicialmente previstos se tornam "sem base ou fundamento". **"Incrementalidade"**: não tentar construir o sistema todo de uma só vez; o sistema é partido em incrementos que podem ser desenvolvidos em paralelo em tempos diferentes, fazendo teste de unidade para cada incremento; quando o incremento é completado e testado, ele é integrado ao sistema. **Convergência**: estabelece que se esteja ativamente atacando todos os riscos que devem ser considerados; como resultado o sistema se torna mais próximo da realidade buscada a cada iteração; na medida em que os riscos são atacados de forma pró-ativa o sistema está sendo entregue em incrementos; faz-se o possível para garantir o sucesso do modo mais rápido possível. **Orientação a pessoas**: favorecer (valorizar) pessoas sobre processos e tecnologias; processos ágeis evoluem através de adaptação, de maneira orgânica; desenvolvedores são encorajados a aumentar sua produtividade, qualidade e desempenho; **"Colaboratividade"**: processos ágeis apóiam comunicação entre membros da equipe, como parte vital de qualquer projeto de desenvolvimento de software; quando um projeto é dividido em partes, entender como as partes se acomodam é vital na criação do produto; há mais integração do que simples comunicação; "colaboratividade" é o ingrediente fundamental para integrar rapidamente um grande projeto à medida em que os incrementos são desenvolvidos em paralelo;
6	**Autor, Ano de publicação**	Aoyama, Mikio. 1998
	Titulo	Agile Software Process and Its Experience.
	Fonte	Proceedings of the 1998 International Conference on Software Engineering, p. 3-12
	Propriedades	**"Incrementalidade" e Evolutividade**: produtos são entregues

	ou Características	incrementalmente ao longo do tempo; **Agilidade**: adaptações rápidas a mudanças nos requisitos e no ambiente; **Modularidade e "Enxutez" ou "Leveza" (*Leanness*):** mais modular e mais leve que modelos de processo convencionais. **Baseado no Tempo**: operado com um ciclo de tempo (prazo) fixo; grandes volumes de desenvolvimento são quebrados em múltiplas entregas que possam ser desenvolvidas incremental e concorrentemente de modo previsível.
7	**Autor, Ano de publicação**	Coram, Michael. Bohner, Shawn. 2005
	Titulo	The Impact of Agile Methods on Software Project Management
	Fonte	Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05)
	Propriedades ou Características	**"Colaboratividade"**: ambiente altamente colaborativo, viabilizando a comunicação informal entre membros da equipe para disseminar informação. **Revisões de Código**: como forma de permitir disseminação de informações chave (ex. através de programação por pares). **Equipes pequenas**: e pequeno número de equipes por projeto, necessário para promover o ambiente colaborativo e por requerer menos planejamento para coordenar as atividades dos membros das equipes. **Cronogramas curtos para entregas**: de 2 semanas a no máximo 6 meses; no fim do prazo um produto funcional é entregue ao cliente para avaliação, com mudanças a serem consideradas na escala de prioridades para as entregas subsequentes. **Delimitação do tempo**: o tamanho das entregas é fixado, mas as funcionalidades não, mantendo o foco no cliente e evitando desperdícios e desvios de escopo. **Testes constantes**: para prevenir a degradação da qualidade, devido entregas muito curtas, dá-se alta ênfase a testes do produto ao longo do ciclo de vida. Visa prevenir também contra a idéia de apenas escrever código. Métodos ágeis requerem testes de integração ao longo do processo de desenvolvimento. Automação dos testes é importante para que as "*builds*" diárias passem por testes de regressão garantindo que todas as funcionalidades estejam operando corretamente. Algumas metodologias ágeis recomendam até mesmo testes de aceitação a cada entrega, ou mesmo em desenvolvimento concorrente.
8	**Autor, Ano de publicação**	Hansson, C. Dittrich, Y. Gustafsson, B. Zarnak, S. 2006
	Titulo	How agile are industrial software development practices?
	Fonte	The Journal of Systems and Software 79, 1295–1311
	Propriedades ou Características	**Valorizar mais Indivíduos e Interações do que Processos e Ferramentas**: a comunicação e a cooperação dentro das equipes de desenvolvimento são consideradas fundamentais e necessárias; um dos argumentos básicos a favor de métodos ágeis é o aumento da eficiência das pessoas que trabalham juntas; interações freqüentes entre indivíduos

		compensam a documentação mais reduzida; um bom processo não conserta uma equipe ruim, mas uma equipe unida e coesa pode trabalhar com um processo ruim; a organização e a ênfase dada à comunicação podem ser usadas como indicadores da agilidade das práticas de desenvolvimento de software. **Valorizar mais Software Funcionando do que Documentação Abrangente**: de acordo com a idéia de desenvolvimento ágil, o sistema funcionando é o melhor meio para avaliar o desempenho da equipe; se a documentação estiver desatualizada, ela é pouco útil, assim, a recomendação ágil é não produzir documentação a não ser que a necessidade delas seja imediata e significativa; a atitude das organizações para com a documentação, e o seu propósito, são indicadores da agilidade das práticas de desenvolvimento de software. **Valorizar mais a Colaboração do Cliente do que Negociação de Contratos**: o manifesto ágil enfatiza que o software não pode ser desenvolvido com sucesso sem o envolvimento dos clientes e/ou usuários; o cliente, ou mais provavelmente, o usuário, conhece as necessidades e os requisitos para o novo software e deve portanto, tomar parte ativa no processo de desenvolvimento; a realimentação (*feedback)* do cliente/usuário deve acontecer de forma regular e freqüente; contratos devem descrever e estipular como a equipe de desenvolvimento e o cliente/usuário trabalharão juntos; uma contrato que especifica apenas requisitos técnicos, prazos e custos, seria falho; um amplo escopo de atividades que normalmente não são consideradas parte do desenvolvimento de software, como a função de apoio ou suporte, estão relacionadas com o envolvimento do usuário e são indicadores da agilidade das práticas de desenvolvimento de software. **Valorizar mais a Resposta a Mudanças do que Seguir um Plano**: de acordo com o movimento ágil, os planos devem ser flexíveis, permitindo respostas a mudanças nos negócios e na tecnologia; a construção de planos detalhados para um horizonte de poucas semanas é útil; elaborar planos de mais alto nível de abstração para os próximos poucos meses e apenas algumas idéias para o futuro é considerada uma boa estratégia; provavelmente, o cliente/usuário irá alterar os requisitos quando ele testar o sistema; novos requisitos serão descobertos ou requisitos já elicitados serão considerados desnecessários durante o processo de desenvolvimento, devendo o plano ser revisto e as prioridades reconsideradas; a grande questão é como lidar com mudanças inevitáveis ao longo do ciclo de vida dos projetos e responder a mudanças no ambiente? O modo pelo qual as organizações reagem a mudanças também é um indicador da agilidade das práticas de desenvolvimento de software.
9	**Autor, Ano de publicação**	Cockburn, A. 2002
	Titulo	Agile Software Development Joins the “Would-Be” Crowd
	Fonte	Cutter IT Journal v. 15 n. 2
	Propriedades ou	**Utilização de subprojetos curtos**: subprojetos de períodos mais longos tornam a equipe menos ágil; a escolha tem que ser equilibrada porque

	Características	há trabalho de planejamento no início e de integração, teste e entrega no fim de cada subprojeto ou incremento. **Reflexão sobre as práticas empregadas**: reuniões no fim de cada subprojeto ou iteração para os membros da equipe discutirem o que eles estão fazendo bem e o que precisa ser mudado. **Trabalho em locais próximos**: para algumas metodologias significa trabalhar na mesma sala ou em salas adjacentes, o que só funciona para equipes de 8 a no máximo 14 pessoas Todas as metodologias são sensíveis à localização da equipe, pois estão fortemente fundamentadas em canais de comunicação rápidos e ricos, que permitem reduzir a documentação externa a ser construida e mantida. A agilidade pode variar variando-se a qualidade dos canais de comunicação e o balanço entre conhecimento externo e tácito. Não significa que a equipe para ser ágil não possa produzir documentos; executar um projeto entre paises diferentes e fusos horários diferentes é realmente um obstáculo para as equipes serem ágeis. **Atenção às Equipes**: as reuniões diárias em pé e os workshops de reflexão dão às pessoas a chance de manifestar suas preocupações com relação ao grupo; os membros da equipe devem ter meios de expressar seus medos e desejos para a comunidade.
10	**Autor, Ano de publicação**	Abrahamsson, P. Salo, O. Ronkainen, J. Warsta, J. 2002
	Titulo	Agile Software Development Methods. Review and Analysis.
	Fonte	Espoo. VTT Publications 478
	Propriedades ou Características	**Modularidade:** no nível de processo de desenvolvimento; **Iteratividade:** ciclos curtos que permitam verificações e correções rápidas; **Limitação de Tempo:** ciclos de iteração de uma a seis semanas; **Parcimônia:** remoção de todas as atividades desnecessárias no processo de desenvolvimento; **Adaptabilidade:** para com possíveis novos riscos emergentes; **"Incrementalidade":** abordagem de processo que permite a construção de funcionalidades da aplicação em pequenos passos; *releases* pequenas com ciclos rápidos. **Convergência:** abordagem para minimizar os riscos; **Orientação a pessoas:** favorecimento das pessoas sobre processos e tecnologia; **"Colaboratividade":** estilo de trabalho comunicativo. **"Cooperatividade":** cliente e desenvolvedores trabalhando constantemente juntos e comunicando-se de perto. **Transparência ou Clareza**: o método é fácil de aprender e modificar e é bem documentado.
11	**Autor, Ano de publicação**	Lindvall, M. Basili, V. et al. 2002
	Titulo	Empirical Findings in Agile Methods,
	Fonte	In: Proceedings of Extreme Programming and Agile Methods – SP/Agile Universe, pp. 197-207.

	Propriedades ou Características	**Iteratividade**: entrega um sistema completo logo no início e então muda a funcionalidade de cada subsistema com cada nova *release*. **"Incrementalidade":** o sistema como especificado nos requisitos é particionado em pequenos subsistemas por funcionalidade; nova funcionalidade é adicionada a cada nova *release*. **Auto-organização:** a equipe tem autonomia para se organizar da melhor forma para completar os itens de trabalho. **Emergência**: permite-se que tecnologia e requisitos aflorem ao longo do ciclo de vida do produto.

5- Análise dos Resultados Obtidos

Foi montada uma tabela de características de métodos ágeis encontradas nos artigos. Esta tabela foi construida a partir de cada característica distinta identificada no conjunto de artigos. Foram contadas as quantidades de artigos em que a característica foi abordada, sendo que nesta contagem cada *cluster* foi computado uma única vez.

Cada *cluster* ficou com 2 documentos, sendo a formação de cada um deles conforme se segue:

Tabela 2 – Formação dos *clusters* de documentos incluidos na revisão sistemática

Cluster	Documento Incluído	Documento Representativo do *Cluster* (ord. na tab. 1)
1	1- Abrahamsson, P.; Warsta, J.; Siponen, M.T. & Ronkainen, J. New directions on agile methods: a comparative analysis. IEEE Computer Society, 2003, 244-254	1
	10- Abrahamsson, P. Salo, O. Ronkainen, J. Warsta, J. Agile Software Development Methods. Review and Analysis. Espoo. VTT Publications 478, 2002	
2	2- Boehm, B. & Turner, R. Balancing agility and discipline: A Guide for the Perplexed. Pearson Education Inc, Boston, MA. 2004a	2
	11- Lindvall, M. Basili, V. et al. Empirical Findings in Agile Methods, In: Proceedings of Extreme Programming and Agile Methods – SP/Agile Universe, pp. 197-207, 2002.	

Segue a tabela com a incidência das características de métodos ágeis nos artigos:

Tabela 3 – Incidência das características de métodos ágeis nos artigos

Característica	Descrição	Quant. Artigos	Ordem na Tab. 1
"Incrementalidade":	Pequenas *releases* de software, com ciclos rápidos de desenvolvimento; não entregar o produto todo de uma só vez;	5	1,2, 5, 6, 9

	não tentar construir o sistema todo de uma só vez; o sistema é partido em incrementos que podem ser desenvolvidos em paralelo em tempos diferentes, fazendo teste de unidade para cada incremento; quando o incremento é completado e testado, ele é integrado ao sistema; o sistema como especificado nos requisitos é particionado em pequenos subsistemas por funcionalidade; nova funcionalidade é adicionada a cada nova *release*. Subprojetos de períodos mais longos tornam a equipe menos ágil.		
"Cooperatividade":	Interação com proximidade entre desenvolvedor e cliente; interação aberta entre os vários *stakeholders* (gerentes, desenvolvedores e clientes); o manifesto ágil enfatiza que o software não pode ser desenvolvido com sucesso sem o envolvimento dos clientes e/ou usuários; o cliente, ou mais provavelmente, o usuário, conhece as necessidades e os requisitos para o novo software e deve, portanto, tomar parte ativa no processo de desenvolvimento; a realimentação (*feedback)* do cliente/usuário deve acontecer de forma regular e freqüente; um amplo escopo de atividades que normalmente não são consideradas parte do desenvolvimento de software, como a função de apoio ou suporte, estão relacionadas com o envolvimento do usuário e são indicadores da agilidade das práticas de desenvolvimento de software.	3	1, 3, 8
Transparência ou Clareza:	O método é fácil de aprender e modificar e é suficientemente documentado.	1	1
Adaptabilidade:	Habilidade de atender e reagir a mudanças de última hora; adaptação do processo para atender situações ou riscos não previstos inicialmente; capacidade de adaptações rápidas a mudanças nos requisitos e no ambiente; de acordo com o movimento ágil, os	5	1, 4, 5, 6, 8

	planos devem ser flexíveis, permitindo respostas a mudanças nos negócios e na tecnologia; provavelmente, o cliente/usuário irá alterar os requisitos quando ele testar o sistema; novos requisitos serão descobertos ou requisitos já elicitados serão considerados desnecessários durante o processo de desenvolvimento, devendo o plano ser revisto e as prioridades reconsideradas. O modo pelo qual as organizações reagem a mudanças também é um indicador da agilidade das práticas de desenvolvimento de software.		
Iteratividade:	Envolvendo vários ciclos curtos, dirigidos por características do produto, nos quais um certo conjunto de atividades é completado em poucas semanas; estes ciclos não serão suficientes para obter os elementos 100% prontos, por isso são repetidos muitas vezes para refinar as entregas.	4	2, 4, 5, 7
Auto-organização:	As equipes determinam o melhor modo de trabalhar; a equipe tem autonomia para se organizar da melhor forma para completar os itens de trabalho.	1	2
Emergência:	Os processos, princípios, estruturas de trabalho são reconhecidos durante o projeto ao invés de serem pré-determinados; permite-se que tecnologia e requisitos emerjam ao longo do ciclo de vida do produto;	1	2
Períodos de reflexão e introspecção;	Reuniões no fim de cada subprojeto ou iteração para os membros da equipe discutirem o que eles estão fazendo bem e o que precisa ser mudado.	2	4, 9
Incorporação de realimentação (*feedback)* rápida;	Equipes capazes de procurar e receber continuamente, realimentação (*feedback)* de modo mais freqüente e com mais rapidez. (teoria dos sistemas adaptativos complexos)	2	3, 4
Modularidade:	Característica que permite que um processo seja quebrado em componentes chamados de atividades; modularidade permite que atividades sejam adicionadas ou removidas de um	2	5, 6

	processo quando necessário.		
Restrição de Prazo:	Estabelecimento de limite de tempo para cada iteração programada; provavelmente não será possível programar todas as atividades do processo numa iteração, mas apenas aquelas para alcançar os objetivos estabelecidos no início da iteração; funcionalidades podem ser reduzidas e atividades reprogramadas se não puderem ser completadas no espaço de tempo alocado para aquela iteração. Opera-se com um ciclo de tempo (prazo) fixo; grandes volumes de desenvolvimento são quebrados em múltiplas entregas que possam ser desenvolvidas incremental e concorrentemente de modo previsível. Tamanho das entregas é fixado, mas as funcionalidades não, mantendo o foco no cliente e evitando desperdícios e desvios de escopo.	3	5, 6, 7
Parcimônia ("Enxutez" ou "Leveza"(*Leanness*):	Característica que o processo ágil tem de requerer o mínimo necessário de atividades para mitigar riscos e alcançar metas; a redução de atividades permitiria aos desenvolvedores entregar sistemas com cronogramas agressivos; de acordo com a idéia de desenvolvimento ágil, o sistema funcionando é o melhor meio para avaliar o desempenho da equipe; se a documentação estiver desatualizada, ela é pouco útil, assim, a recomendação ágil é não produzir documentação a não ser que a necessidade delas seja imediata e significativa; a atitude das organizações para com a documentação, e o seu propósito, são indicadores da agilidade das práticas de desenvolvimento de software; remoção de todas as atividades desnecessárias no processo de desenvolvimento;	3	5, 6, 8
Convergência:	Ataque efetivo a todos os riscos que devem ser considerados; como resultado o sistema se torna mais próximo da realidade buscada a cada iteração; à medida que os riscos são	1	5

	atacados de forma pró-ativa, o sistema está sendo entregue em incrementos; faz-se o possível para garantir o sucesso do modo mais rápido possível.		
Orientação a pessoas:	Favorecer (valorizar) pessoas sobre processos e tecnologias; processos ágeis evoluem através de adaptação, de maneira orgânica; desenvolvedores são encorajados a aumentar sua produtividade, qualidade e desempenho; a comunicação e a cooperação dentro das equipes de desenvolvimento são consideradas fundamentais e necessárias; um dos argumentos básicos a favor de métodos ágeis é o aumento da eficiência das pessoas que trabalham juntas; a organização e a ênfase dada à comunicação podem ser usadas como indicadores da agilidade das práticas de desenvolvimento de software. As reuniões diárias em pé e os workshops de reflexão dão às pessoas a chance de manifestar suas preocupações com relação ao grupo; os membros da equipe devem ter meios de expressar seus medos e desejos para a comunidade.	3	5, 8, 9
"Colaboratividade":	Processos ágeis apoiam comunicação entre membros da equipe, como parte vital de qualquer projeto de desenvolvimento de software; quando um projeto é dividido em partes, entender como as partes se acomodam é vital na criação do produto; há mais integração do que simples comunicação; colaboração é o ingrediente fundamental para integrar rapidamente um grande projeto à medida em que os incrementos são desenvolvidos em paralelo; ambiente altamente colaborativo, viabilizando a comunicação informal entre membros da equipe para disseminar informação.	3	4, 5, 7
Equipes pequenas:	Pequeno número de equipes por projeto, necessário para promover o ambiente colaborativo e por requerer menos planejamento para coordenar as	1	7

	atividades dos membros das equipes.		
Testes constantes:	Para prevenir a degradação da qualidade devido a entregas muito curtas, dá-se alta ênfase a testes do produto ao longo do ciclo de vida. Visa prevenir também contra a idéia de apenas escrever código. Métodos ágeis requerem testes de integração ao longo do processo de desenvolvimento. Automação dos testes é importante para que as "*builds*" diárias passem por testes de regressão garantindo que todas as funcionalidades estejam operando corretamente. Algumas metodologias ágeis recomendam até mesmo testes de aceitação a cada entrega, ou mesmo em desenvolvimento concorrente.	1	7
Equipes Locais:	Para algumas metodologias significa trabalhar na mesma sala ou em salas adjacentes, o que só funciona para equipes de 8 a no máximo 14 pessoas Todas as metodologias são sensíveis à localização da equipe, pois estão fortemente fundamentadas em canais de comunicação rápidos e ricos, que permitem reduzir a documentação externa a ser construida e mantida. A agilidade pode variar variando-se a qualidade dos canais de comunicação e o balanço entre conhecimento externo e tácito. Não significa que a equipe para ser ágil não possa produzir documentos; executar um projeto entre paises diferentes e fusos horários diferentes é realmente um obstáculo para as equipes serem ágeis.	1	9

Dentre os artigos ou documentos analisados, 11 apresentaram caracterizações de agilidade para métodos de desenvolvimento de software. Nestes artigos foram encontradas diversas características com significados similares, apesar de denominações diferentes.

Conforme a tabela 1, os 11 artigos são:

1- Abrahamsson, P.; Warsta, J.; Siponen, M.T. & Ronkainen, J. New directions on agile methods: a comparative analysis. IEEE Computer Society, 2003, 244-254

2- Boehm, B. & Turner, R. Balancing agility and discipline: A Guide for the Perplexed. Pearson Education Inc, Boston, MA. 2004a.

3- Meso, Peter. Jain, Radhika. Contemporary practices in systems development. Agile Software Development: adaptive systems principles and best practices. Information Systems Management, summer 2006.

4- Holmstrom, Helena. Fitzgerald, Brian. et al. Contemporary practices in systems development. Agile Practices Reduce Distance in Global Software Development. Information Systems Management, summer 2006.

5- Miller, Granville G. The Characteristics of Agile Software Processes. Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01), 2001.

6- Aoyama, Mikio. Agile Software Process and Its Experience. Proceedings of the 1998 International Conference on Software Engineering, p. 3-12, 1998

7- Coram, Michael. Bohner, Shawn. The Impact of Agile Methods on Software Project Management. Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 2005

8- Hansson, C. Dittrich, Y. Gustafsson, B. Zarnak, S. How agile are industrial software development practices? The Journal of Systems and Software 79, 1295–1311, 2006

9- Cockburn, A. Agile Software Development Joins the "Would be" Crowd. Cutter IT Journal, v.15, n.2, 2002.

10- Abrahamsson, P. Salo, O. Ronkainen, J. Warsta, J. Agile Software Development Methods. Review and Analysis. Espoo. VTT Publications 478, 2002

11- Lindvall, M. Basili, V. et al. Empirical Findings in Agile Methods, In: Proceedings of Extreme Programming and Agile Methods – SP/Agile Universe, pp. 197-207, 2002.

Após interpretação do significado descrito para cada ocorrência de característica, foram identificadas as similaridades e contadas às respectivas presenças nos documentos ou artigos incluídos na revisão sistemática. As revisões de código consideradas por Coram e Bohner (2005) foram interpretadas mais como prática do que característica, razão pela qual foi excluida.

A tabela 3 mostra a distribuição das 18 características encontradas, de acordo com a incidência de presença em 9 artigos após a exclusão das similaridades e consideração dos clusters.

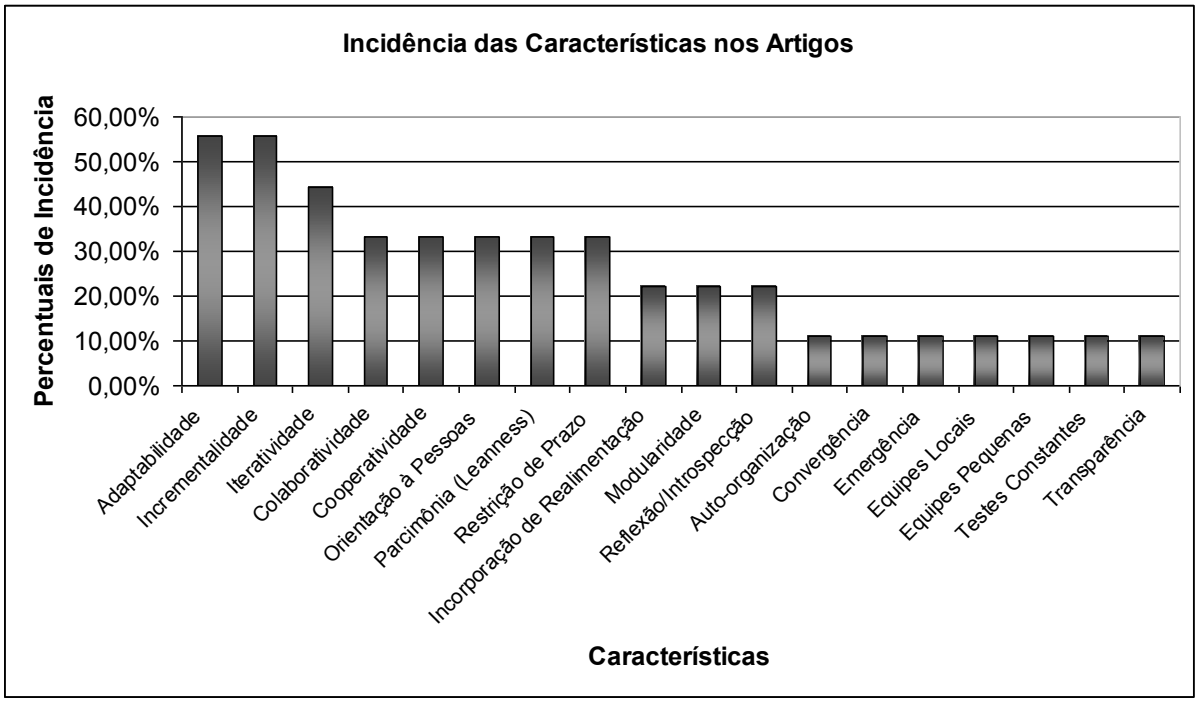
Tabela 3 – Percentuais de incidência das características nos artigos

Ord	Característica	Quantidade de Artigos	Percentual
1	Adaptabilidade	5	55,56%
2	"Incrementalidade"	5	55,56%

3	Iteratividade	4	44,44%
4	"Colaboratividade"	3	33,33%
5	"Cooperatividade"	3	33,33%
6	Orientação a Pessoas	3	33,33%
7	Parcimônia (Leanness)	3	33,33%
8	Restrição de Prazo	3	33,33%
9	Incorporação de Realimentação	2	22,22%
10	Modularidade	2	22,22%
11	Reflexão/Introspecção	2	22,22%
12	Auto-organização	1	11,11%
13	Convergência	1	11,11%
14	Emergência	1	11,11%
15	Equipes Locais	1	11,11%
16	Equipes Pequenas	1	11,11%
17	Testes Constantes	1	11,11%
18	Transparência	1	11,11%

Para facilitar a visualização, foi elaborado o gráfico que se segue, a partir da tabela 3:

Gráfico 1 – Percentuais de incidência das características nos artigos



Como se pode observar, as características mais freqüentes foram Adaptabilidade, "Incrementalidade" e Iteratividade, com 55,56 % de incidência nos artigos para as duas primeiras e 44,44 % para a Iteratividade. Dentre estas, a adaptabilidade parece estar mais diretamente alinhada com um dos valores do Agile Manifesto (2001): " Valorizar mais a resposta a mudanças do que seguir um plano".

As características com uma frequência média de presença nos artigos foram "Colaboratividade", "Cooperatividade", Orientação à Pessoas, Parcimônia e Restrição de Prazo. Dentre estas, parecem estar mais alinhadas com valores especificados no Agile Manifesto (2001):

"Cooperatividade" – associada com "Valorizar mais a ação do cliente do que negociação de contratos".

Orientação à Pessoas – associada com "Valorizar mais indivíduos e interações do que processos e ferramentas".

Parcimônia – associada com "Valorizar mais software funcionando do que documentação abrangente".

A "parcimônia" foi um termo encontrado na literatura [Miller, 2001] e adotado nesta revisão para "*leanness*". Segundo Conboy e Fitzgerald (2004) a idéia de "*leanness*" é a eliminação de perdas ou a habilidade de fazer mais com menos.

As características menos frequentes nos artigos foram Incorporação de Realimentação (*feedback)*, Modularidade, Reflexão/Introspecção, Auto-organização, Convergência, Emergência, Equipes Locais, Equipes Pequenas, Testes Constantes e Transparência.

Dentre as 18 características identificadas, três foram contempladas por 2 clusters (("incrementalidade", adaptabilidade e iteratividade), dez foram contempladas por apenas 1 cluster ("cooperatividade", transparência, auto-organização, emergência, modularidade, restrição de prazo, parcimônia, convergência, orientação a pessoas e "colaboratividade") e 5 não têm participação em qualquer cluster (reflexão e introspecção, realimentação (*feedback)* rápida, equipes pequenas e equipes locais). A tabela a seguir mostra a distribuição das características identificadas nos artigos, de acordo com os *clusters* a elas associados.

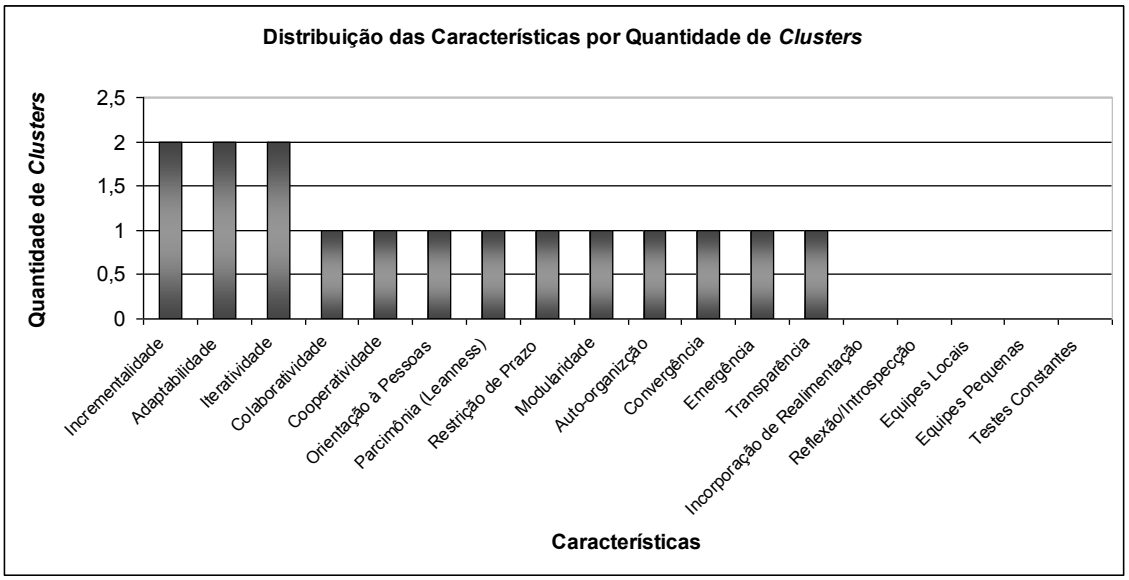
Tabela 4 - Distribuição das características identificadas nos artigos, por quantidade de *clusters*

Ord	Característica	Quantidade de *Clusters*	Quantidade de Artigos
1	"Incrementalidade"	2	5
2	Adaptabilidade	2	5
3	Iteratividade	2	4
4	"Colaboratividade"	1	3
5	"Cooperatividade"	1	3
6	Orientação à Pessoas	1	3
7	Parcimônia (Leanness)	1	3
8	Restrição de Prazo	1	3
9	Modularidade	1	2
10	Auto-organização	1	1
11	Convergência	1	1
12	Emergência	1	1

13	Transparência	1	1
14	Incorporação de Realimentação	0	2
15	Reflexão/Introspecção	0	2
16	Equipes Locais	0	1
17	Equipes Pequenas	0	1
18	Testes Constantes	0	1

Para facilitar a visualização, foi elaborado o gráfico que se segue, a partir da tabela 4:

Gráfico 2 – Percentuais de incidência das características nos clusters



Observa-se que as características de "Incrementalidade", Adaptabilidade e Iteratividade permanecem com maior destaque também quando se considera a quantidade de *clusters* associados às características identificadas nos artigos.

Já as características de Auto-organização, Convergência, Emergência e Transparência que estavam na faixa de baixa incidência nos artigos, tiveram um *cluster* associado a cada uma delas.

A tabela 5 abaixo dá uma idéia da distribuição das características abordadas nos artigos por faixa de ano de publicação.

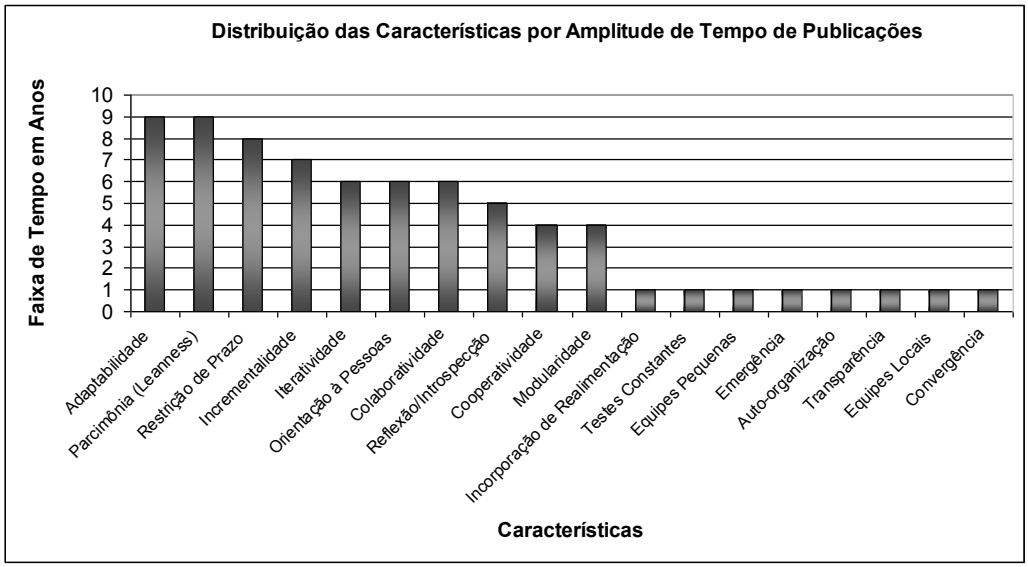
Tabela 5 - Distribuição das características por faixa de ano de publicação dos artigos

Ord	Característica	Faixa		Ampli-tude	Quant. Artigos	Quant. por ano	Obs.
		De	Até				
1	Adaptabilidade	1998	2006	9	5	**0,56**	2 em 2006
2	Parcimônia (*Leanness*)	1998	2006	9	3	0,33	
3	Restrição de Prazo	1998	2005	8	3	0,38	
4	"Incrementalidade"	1998	2004	7	5	**0,71**	
5	Iteratividade	2001	2006	6	4	**0,67**	
6	Orientação à Pessoas	2001	2006	6	3	0,50	
7	"Colaboratividade"	2001	2006	6	3	0,50	
8	Reflexão/Introspecção	2002	2006	5	2	0,40	
9	"Cooperatividade"	2003	2006	4	3	**0,75**	2 em 2006
10	Modularidade	1998	2001	4	2	0,50	
11	Incorporação de Realimentação	2006	2006	1	2	2,00	2 em 2006
12	Testes Constantes	2005	2005	1	1	1,00	
13	Equipes Pequenas	2005	2005	1	1	1,00	
14	Emergência	2004	2004	1	1	1,00	
15	Auto-organização	2004	2004	1	1	1,00	
16	Transparência	2003	2003	1	1	1,00	
17	Equipes Locais	2002	2002	1	1	1,00	
18	Convergência	2001	2001	1	1	1,00	

Na tabela 5 pode-se observar que dentre as características abordadas com maior amplitude de tempo e com índice de publicações por ano acima de 0,50 artigos estão a adaptabilidade, a "incrementalidade", a iteratividade e a "cooperatividade". Também se podem observar na tabela, que algumas características foram preocupações apenas pontuais por parte dos especialistas.

O gráfico que se segue foi elaborado a partir da tabela 5, oferecendo uma visualização das características por amplitude de tempo de publicações:

Gráfico 3 – Distribuição das Características por Amplitude de Tempo de Publicações



Observa-se também, que com exceção do artigo publicado por Aoyama (1998) todos os demais foram publicados a partir da declaração do Agile Manifesto (2001).

6- Proposta de Caracterização Básica para Métodos Ágeis

Os métodos ágeis têm como marco principal a publicação do Agile Manifesto (2001) e procuram colocar mais ênfase nas pessoas e interações, software funcionando, colaboração do cliente e acomodação das mudanças, ao invés de focar mais nos processos, ferramentas, contratos e planos.

Todas as características de métodos ágeis elencadas nesta revisão podem ser desejáveis em ambientes específicos de desenvolvimento de software, no sentido de que podem trazer benefícios quanto ao melhor desempenho do processo.

Não seria adequado afirmar que um determinado método para ser considerado ágil tenha, necessariamente, que conter integralmente todas as características apresentadas. Algumas das características levantadas podem estar presentes em determinado método com grau de intensidade maior ou menor, ou até mesmo algumas delas não estarem presentes, dependendo de fatores circunstanciais, sem comprometer a agilidade. O foco para avaliação tem que ser o resultado final alcançado, no sentido de satisfazer necessidades do cliente, dentro do orçamento, dentro dos prazos e com a qualidade esperada. Além disso, não necessariamente as diversas instâncias possíveis de métodos ágeis têm o mesmo grau de agilidade.

Uma idéia alternativa para configurar o método com o nível ou grau de agilidade desejado, é a adoção de práticas. As práticas adotadas podem ser ajustadas de modo a alcançar a intensidade desejada e adequada para cada característica que deve estar presente no método ágil. Desse modo, poder-se-ia tentar embutir nele, com alguma intensidade, algumas ou todas as características dos métodos ágeis. Métodos tradicionais ou orientados a plano podem ser customizados com uma dosagem de algumas práticas para alcançar o nível de agilidade necessário. Segundo Patel et al (2006) a tendência de adaptação/customização dos processos de desenvolvimento de software por parte das organizações desenvolvedoras está em crescimento, enquanto

decresce a quantidade de organizações que seguem um processo comercial de modo prescritivo.

Os fatores que podem influenciar a escolha das práticas e seus respectivos ajustes, para serem inseridas no processo de software e alcançar a agilidade necessária para o método de desenvolvimento devem contemplar aspectos como natureza do projeto, ambiente de desenvolvimento, cultura da organização.

Para se alcançar a agilidade de métodos de desenvolvimento de software, algumas características podem ser consideradas fundamentais. Dentro do contexto apresentado, a característica fundamental e mais aderente aos valores do Manifesto Ágil parece ser a adaptabilidade. Em seguida, e considerando um aspecto mais pragmático, seguem as características de incrementalidade e iteratividade que devem caminhar juntas. A essas características devem se somar outras três que estão diretamente associadas uma idéia central dos métodos ágeis que é o foco nas pessoas que conduzem as atividades dos processos. São elas: "colaboratividade", "cooperatividade" (estas duas características às vezes são confundidas uma com a outra na literatura) e orientação a pessoas. Deve-se acrescentar ainda, duas características que além de necessárias para o sucesso de um método ágil, são fundamentais em qualquer processo produtivo inserido em uma economia global e um mercado cada vez mais competitivo. São elas: parcimônia (*leanness*) e restrição de prazo. Convém observar que as entregas devem ser feitas com rapidez, mas têm que agregar valor para o cliente, através de resultados práticos ou efetivos, não devendo ser confundidas com a prática de prototipação.

As características de "colaboratividade" e "cooperatividade" às vezes são confundidas uma com a outra na literatura, dependendo da interpretação que se faça para o seu significado. Contudo, nesta *quasi*-revisão sistemática não foi feita a mescla destas duas características, tentando respeitar as idéias dos autores, além de não se ter encontrado indicativos que permitissem fazer tal mescla de forma segura. A diferença entre as duas é sutil e não está destacada nos textos dos artigos recuperados. A idéia da cooperação parece ter ligação mais direta com a interação do cliente com o grupo desenvolvedor e está associada ao papel da realimentação (*feedback)* constante [Abrahamsson et al, 2003; Meso e Radhika, 2006; Hansson et al, 2006]. Já a idéia de "colaboratividade" parece estar associada com a interação entre os membros do grupo desenvolvedor e relacionada com a integração contínua de novos incrementos de software, com funcionalidades modificadas ou novas [Holmstrom et al, 2006; Miller, 2001; Coram e Bohner, 2005; Miller, 2003].

As demais características elencadas nesta *quasi*-revisão sistemática são também importantes para o sucesso de um método ágil. Contudo, pode-se considerá-las como decorrentes ou subsidiárias daquelas acima citadas como fundamentais.

Abrahamsson, Salo, Ronkainen e Warsta (2002) consideram que um método é agil se ele apresenta 4 características: "incrementalidade", "cooperatividade", transparência ou clareza e adaptabilidade. Para eles, o pensamento ágil é uma visão centrada nas pessoas para desenvolver software.

Para Highsmith (2002) as pessoas, com seus conhecimentos, habilidades, experiências, peculiaridades e personalidades são o principal fator ou impacto de primeira grandeza no sucesso de um projeto de software.

Nossa proposta para qualificar um método ou processo de desenvolvimento de software como sendo ágil, é apresentada assim:

um método para ser dito ágil, deve apresentar, em um grau adequado ao contexto de desenvolvimento em que ele se insere, no mínimo as seguintes características: adaptabilidade, incrementalidade, iteratividade, "colaboratividade", "cooperatividade", orientação a pessoas, parcimônia (leanness) e restrição de prazo.

Uma grande questão que fica é: como medir a intensidade ou grau de adequação destas características a um determinado contexto de desenvolvimento de software?

7- Conclusões

A caracterização de agilidade em métodos ágeis, de maneira geral, e não em um ou mais métodos específicos, é importante para o melhor entendimento de outras questões ligadas ao desenvolvimento ágil. A avaliação do grau em que os métodos ágeis existentes atendem aos princípios declarados no manifesto ágil pode ajudar a entender os resultados da aplicação de certos métodos em projetos de desenvolvimento de software.

Para Favaro (2002) a iteratividade é considerada o denominador comum dos métodos ágeis.

Glass (2004) destaca a necessidade de pesquisas que proponham uma taxonomia para as metodologias ágeis disponíveis, baseada em seus pontos fortes e em seus pontos fracos; uma segunda taxonomia, para o espectro de domínios de problemas, baseada em suas necessidades; e um mapeamento entre as duas taxonomias baseado no entendimento do que as metodologias oferecem e do que os domínios necessitam.

A análise do significado ou definição de agilidade a partir do seu relacionamento com flexibilidade e "*leanness*" (parcimônia) pode apoiar o entendimento dos métodos ágeis. Explorar a natureza e evolução destes conceitos em outras áreas (manufatura, finanças, gerência, trabalho, marketing, eletrônica) pode ajudar no esclarecimento do significado de agilidade dos métodos de desenvolvimento de software. Para Lou et al (2004) agilidade tem dois significados na indústria manufatureira moderna: flexibilidade e reconfigurabilidade. Flexibilidade seria a habilidade da organização para fazer ajustes de acordo com a necessidade dos clientes; reconfigurabilidade seria a habilidade da organização para atender demandas por mudanças. A habilidade para responder a mudanças de mercado tem sido reconhecida com um elemento chave para o sucesso e a sobrevivência das organizações.

Conboy e Fitzgerald (2004), após analisar os conceitos de flexibilidade e parcimônia (*leanness*) associados à agilidade, propõem o seguinte conceito: agilidade é a contínua prontidão de um método ou processo ágil, para rapidamente ou inerentemente, proativamente ou reativamente, abraçar a mudança, através de componentes econômicos, simples e de alta qualidade e através de relacionamentos com seu ambiente. Segundo os mesmos autores, a parcimônia (*leanness*) pressupõe a eliminação de todas as perdas em um processo, enquanto que a agilidade requer sim a eliminação de perdas, mas só até o limite que não comprometa a habilidade de pronta resposta a mudanças.

O alinhamento da filosofia dos métodos ágeis com os objetivos do desenvolvimento de software global (GSD – *Global Software Development*), para

resolver questões incidentes nesse ambiente de desenvolvimento pode ser pesquisado, como sugere Holmstrom *et al* (2006), sendo esta uma idéia interessante. Esse alinhamento pode ter uma associação direta com as características de Equipes Locais e Equipes Pequenas.

A busca de um caminho para descobrir associações entre características de métodos ágeis de desenvolvimento de software e características de projetos de desenvolvimento de software, para determinar se métodos ágeis são indicados para os projetos, ainda permanece sem uma resposta satisfatória.

A customização de métodos de desenvolvimento de software, identificando e selecionando as práticas mais adequadas para um dado ambiente ou projeto de desenvolvimento, obtendo uma mescla equilibrada de práticas dos métodos dirigidos a planos, com práticas dos métodos ágeis, parece ainda carecer de mais trabalhos de pesquisa. Segundo Chau *et al* (2002) a criação e compartilhamento de conhecimento são partes essenciais de processos de software, quer sejam tradicionais ou ágeis

Vinekar, Slinkman e Nerur (2006) declaram que há evidências que parecem indicar que as organizações de desenvolvimento de software estão tentando utilizar ambas as abordagens de desenvolvimento de software, a tradicional e a ágil. Para Nawrocki *et al* (2006) a maioria dos projetos de software contemporâneos requerem um equilíbrio de agilidade e disciplina. Como chegar a esse equilíbrio é uma questão que ainda precisa de mais pesquisas, apesar das propostas já existentes feitas por alguns especialistas, dentre eles Boehm e Turner (2004a).

Há ainda que se considerar que os trabalhos de pesquisa podem tomar outros rumos, não necessariamente em alto nível de abstração, podendo se voltar a aspectos mais específicos ligados às atividades dos processos ágeis, como "testes ágeis" em projetos de software. Recentemente, as equipes de desenvolvimento de software que utilizam processos ágeis começaram a adotar largamente o desenvolvimento dirigido a testes – *Test-Driven Development* [Crispin, 2006].

As características relacionadas com testes não tiveram muita incidência nas abordagens feitas nos diversos artigos. Contudo, conforme Lindvall *et al* (2002) enfatizam, os testes são uma questão fundamental em métodos ágeis, fortalecendo a idéia da necessidade de pesquisas nesta direção.

Segundo Barnett (2004) o desenvolvimento *Open Source* adota práticas de planejamento e de engenharia que já provaram ser escaláveis, e que tais práticas são também adotadas pelo desenvolvimento ágil: *releases* freqüentes e mais cedo; representação do cliente no projeto; implementação de TDD – *Test Driven Development.* Este assunto também é merecedor de um pouco mais de pesquisa.

Alguns trabalhos futuros podem se tornar interessantes dentro do contexto de métodos ágeis. Dentre eles:

1- Buscar o entendimento das características de cada método ágil em particular, para identificar o quanto aderente cada um está em relação às características elencadas nesta *quasi*-revisão sistemática.
2- Qual ou quais características estão mais presentes nas metodologias ágeis propostas na literatura e com que intensidade.

3- Qual o significado ou a importância da prática de TDD (*Test-Driven Developement*) na agilidade das metodologias? Para que características dos métodos ágeis TDD contribui, no sentido de tornar o método mais ágil?
4- Procurar métricas para apoiar o estabelecimento e a avaliação do grau de intensidade da agilidade? (possivelmente presença de práticas associadas às diversas características, aplicadas em níveis de intensidade diferentes)
5- Ampliar conhecimento sobre questões como ambientes de desenvolvimento com equipes grandes ou GSD – *Global Software Development*. Estariam tais ambientes impedidos de serem ágeis devido suas maiores dificuldades em atender as características "Equipes Pequenas" e "Equipes Locais"?

8- Referências Bibliográficas

Abrahamsson, P. Salo, O. Ronkainen, J. Warsta, J. (2002) "Agile Software Development Methods. Review and Analysis", Espoo. VTT Publications 478.

Abrahamsson, P.; Warsta, J.; Siponen, M.T. & Ronkainen, J. (2003) "New directions on agile methods: a comparative analysis", IEEE Computer Society, 244-254

Agile Manifesto. (2001) Disponível em http://agilemanifesto.org. Acessado em 23 de Agosto de 2006.

Aoyama, Mikio. (1998) "Agile Software Process and Its Experience", Proceedings of the 1998 International Conference on Software Engineering, p. 3-12.

Barnett, L. (2004) "Applying Open Source Processes In Corportate Development Organizations", Forrester Best Practices, may 20.

Boehm, B. (2002) "Get ready for agile methods, with care", Computer, Institute of Electrical and Electronics Engineers Computer Society, 35, 64-69.

Boehm, B. & Turner. R. (2003) "Observations on balancing discipline and agility", Proceedings of the Agile Development Conference.

Boehm, B. & Turner, R. (2004) "Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods", Institute of Electrical and Electronics Engineers Computer Society, Piscataway, NJ 08855-1331, United States, 26, 718-719.

Boehm, B. & Turner, R. (2004a) "Balancing agility and discipline: A Guide for the Perplexed", Pearson Education Inc, Boston, MA.

Chau, T. Maurer, F. Melnik, G. (2002) "Knowledge Sharing: Agile Methods vs. Tayloristic Methods", Department of Computer Science, University of Calgary Calgary, Canada.

Cockburn, A. (2002) "Agile Software Development Joins the 'Would be' Crowd", Cutter IT Journal, v.15, n.2.

Conboy, K. & Fitzgerald, B. (2004) "Toward a conceptual framework of agile methods: A study of agility in different disciplines", Association for Computing Machinery, New York, NY 10036-5701, United States, 37-44.

Coram, Michael. Bohner, Shawn. (2005) "The Impact of Agile Methods on Software Project Management", Proceedings of the 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05).

Crispin, L. (2006) "Driving Software Quality: How Test-Driven Development Impacts Software Quality", Ieee Software.

Favaro, J. Managing Requirements for Business Value. IEEE Software, v19, 15-17, 2002.

Fowler, M. Highsmith, J. (2001) "The Agile Manifesto", Software Development, v.9 n.8, aug.

Glass, R.L. (2004) "Matching methodology to problem domain", Communications of the ACM, Association for Computing Machinery, New York, United States, 47, 19-21.

Hansson, C. Dittrich, Y. Gustafsson, B. Zarnak, S. (2006) "How agile are industrial software development practices?", The Journal of Systems and Software 79, 1295–1311.

Hazzan, O. Dubinsky, Y. (2006) "Can Diversity in Global Software Development be Enhanced by Agile Software Development?", ACM Digital Library.

Holanda, A. B. (1977) "Dicionário Aurélio da Língua Portuguesa", Editora Nova Fronteira.

Highsmith, J. (2002) "Agile Software Development Ecosystems", Addison Wesley.

Holmstrom, Helena. Fitzgerald, Brian. et al. (2006) "Contemporary practices in systems development. Agile Practices Reduce Distance in Global Software Development", Information Systems Management, summer.

Lello Popular. (1958) "Novo Dicionário Ilustrado da Língua Portuguesa", Lello e Irmão Ed. Porto, Portugal.

Lindvall, M. Basili, V. et al. (2002) "Empirical Findings in Agile Methods", In: Proceedings of Extreme Programming and Agile Methods – SP/Agile Universe, pp. 197-207.

Longman Group. (1992) "Dictionary of Contemporary English – New Edition", Longman House, Burnt Mill, Harlow, UK.

Lou, Ping. Zhou, Zu-de. et al. (2004) "Study on multi-agent-based agile supply chain management", International Journal Adv Manufacturing Tachnology, v.23:197-203.

Meso, Peter. Jain, Radhika. (2006) "Contemporary practices in systems development. Agile Software Development: adaptive systems principles and best practices", Information Systems Management, summer.

Miller, Granville G. (2001) "The Characteristics of Agile Software Processes", Proceedings of the 39th Int'l Conf. and Exhibition on Technology of Object-Oriented Languages and Systems (TOOLS'01).

Nawrocki, J.; Olek, L.; Jasinski, M.; Paliswiat, B.; Walter, B.; Pietrzak, B. & Godek, P. (2006) "Balancing agility and discipline with Xprince", Springer Verlag, Heidelberg, D-69121, Germany, 3943 NCS, 266-277.

Pai, M. McCulloch, M. Gorman, J.D. et al. (2004) "Systematic Reviews and meta-analyses: An illustrated, step-by-step guide", The National Medical Journal of India, vol. 17, n.2.

Patel, C. Lycett, M. et al. (2006) "Perceptions of Agility and Collaboration in Software Development Practice", Proceedings of the 39th Hawaii International Conference on System Sciences.

Rajlich, V. (2006) "Changing the Paradigm of Software Engineering", Communications of the ACM, v.49, n.8, aug.

Sugden, R.C. Strens, M.R. (1996) "Strategies, tatics and methods for handling change", Engineering of Computer-Based Systems – Proceedings of IEEE Symposium and Workshop, mar.

Taylor, P.S. Greer, D. Sage, P. et all. (2006) "Do Agile GSD Experience Reports Help the Practitioner?", ACM Digital Library.

Vinekar, V. Slinkman, C.W. Nerur, S. (2006) "Contemporary practices in systems development. Can Agile and Traditional Systems Development Approaches Coexist? An Ambidextrous View", Information Systems Management, summer.